Projeto para a disciplina Programação Orientada a Objetos

Disciplina: SSC0103-1 - Programação Orientada a Objetos

Professor: Dr. Márcio Delamaro

Monitor PAE: Misael Jr.

Aluno: Emiliano Maia Lobo Menezes

Número USP: 8936871

Introdução

A proposta, visto o caráter individual do meu projeto, foi simples: um sistema de biblioteca funcional, capaz de armazenar informações além do tempo de execução e apresentado ao usuário através de um GUI. Ela apresentou , porém, desafios próprios e nuances particulares.

Foi almejado um sistema que apresenta as funcionalidades esperadas para o gerenciamento do catálogo de uma biblioteca e seus usuários de modo que haja real aplicabilidade do projeto. Também é importante que o projeto seja utilizável não somente pelo desenvolvedor ou programadores terceiros, mas sim qualquer pessoa que esteja familiarizada com o uso de computadores no dia-a-dia. Para isso, o programa funcionará em uma GUI simples e com comandos de fácil entendimento.

Lógica do armazenamento de informação

Antes de começar o trabalho, não estavam claras para mim a prática de *serialization,* ou seja, o armazenamento binário de objetos no sistema em arquivos (do tipo .txt) para ser lido e recuperado em execuções futuras. Quando esses conhecimentos surgiram, funções do trabalho estavam estruturadas e refaze-las não era seguramente viável dentro do prazo. Porém, foi desenvolvida uma lógica de armazenamento de informações própria para o sistema;

Para o sistema, usuários e livros são itens da biblioteca e seguem um padrão de armazenamento. No sistema existem dois arquivos (Usuarios e Livros) cada um contendo as informações dos seus respectivos itens no formato de uma linha no arquivo (cada linha contendo as informações de um item). Por exemplo:

Usuário

Emiliano Maia Lobo Menezes, 34669370; Aluno;0/0/0;*;Harry Potter;10;09/16

Nome; código de identificação; Tipo de usuário; data de penalidade; separador; livro alugado; data de vencimento da locação

Livro

Harry Potter, 3248b; J K rowling;5;*;Emiliano Maia Lobo Menezes;10;09/16

Titulo; código de identificação; Autor; Número de cópias disponíveis; separador; Locador; data de vencimento da locação

Para ambos, existem dois espações principais pré e pós separador. Pré-separador estão os dados fixos do item, ou seja, campos que o itens sempre tem como nome/título e etc. Pós-separador está a lista de livros alugados/locadores e suas respectivas datas de entrega.

Quando operações são feitas no sistema, as linhas requisitadas são lidas, editadas conforme necessário e reescritas. Isso se da através da classe principal do programa; OperadorDeArquivos.

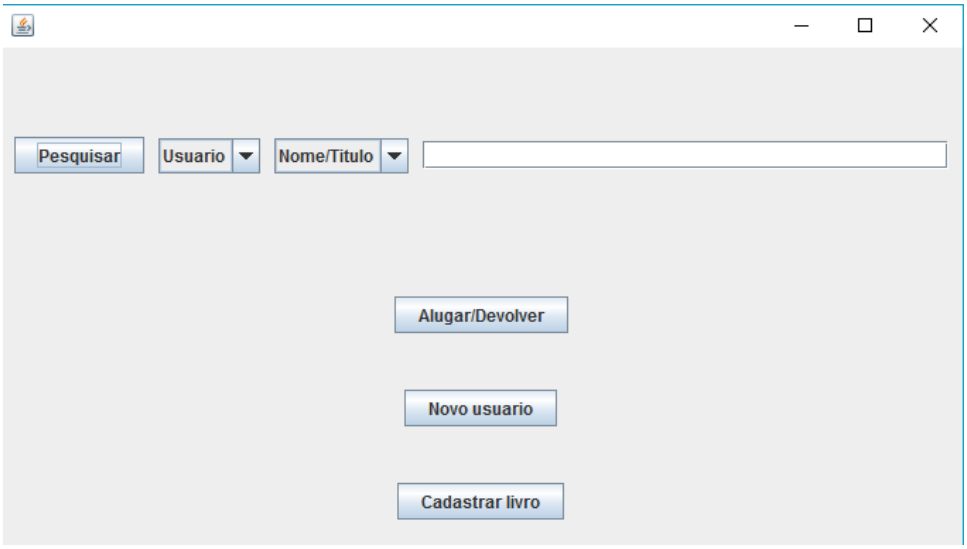
Como o nome pode sugerir, essa é a classe responsável pelos métodos de edição e leitura de arquivos. Ela utiliza a classe *File* do java porém abstraída e com implementações específicas para as demandas do trabalho (para mais informações sobre as métodos da classe por favor consultar o java.doc).

Portanto, OperadorDeArquivos é a classe principal do sistema. A partir dela todas as operações são feitas. Foi primordial para o trabalho que essa classe estivesse bem implementada para suprir as demandas específicas do sistema de maneira confiável e que abstraísse as nuances da escrita e leitura de arquivos a fim de permitir a operabilidade do sistema.

Funcionamento

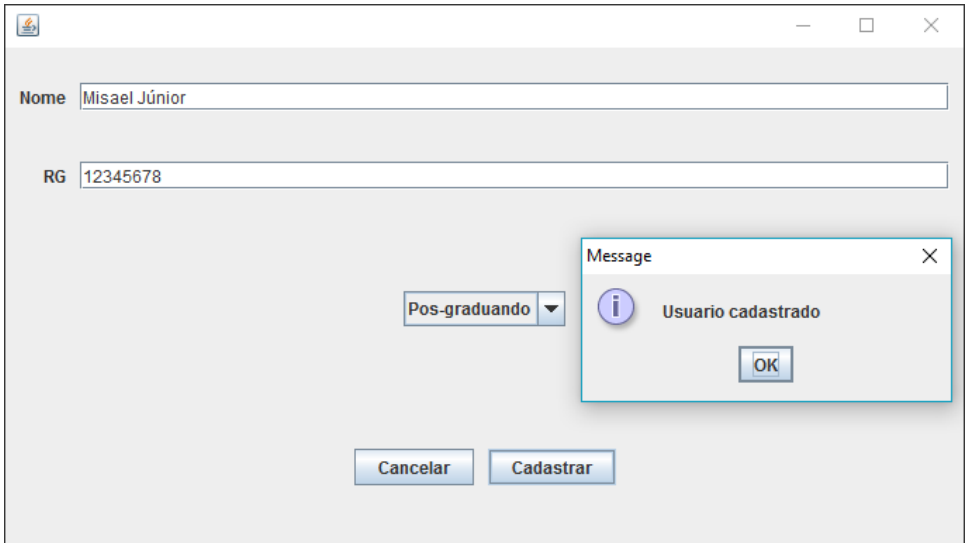
Apesar de em certos momentos apresentar problemas na execução total, a "soma das partes" do sistema, ou seja, os métodos separados, funcionam como esperado de maneira confiável (mesmo os mais complexos da classe OperadorDeArquivos).

O sistema tem três funções principais para operar: Pesquisar, alugar/devolver livro e cadastrar/apagar item.



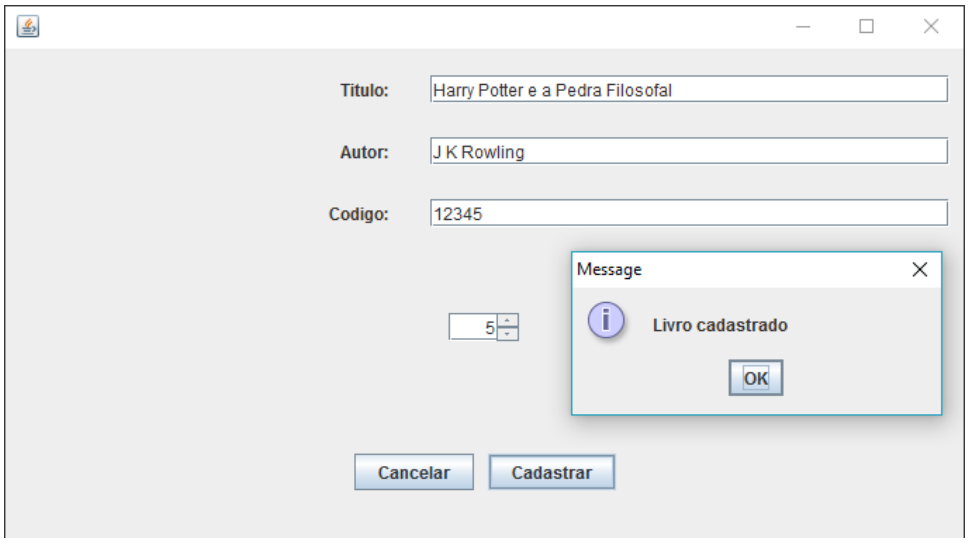
Para intuitos demonstração foi crida um usuário foram feitas as seguintes operações:

Cadastramento de um usuário.



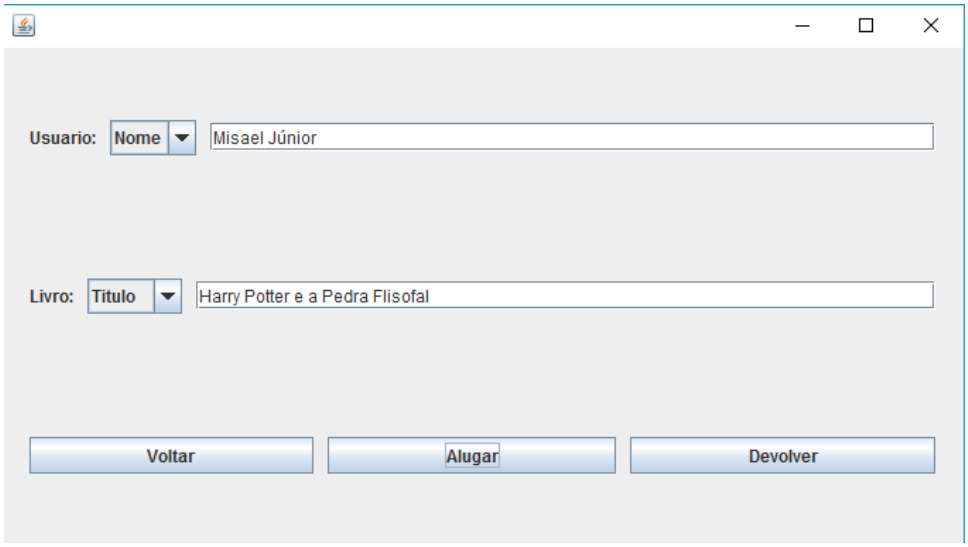
Gerou a linha Misael Júnior;12345678;Pos-graduando;0/0/0;* no arquivo Usuarios.txt.

Cadastramento de um livro.



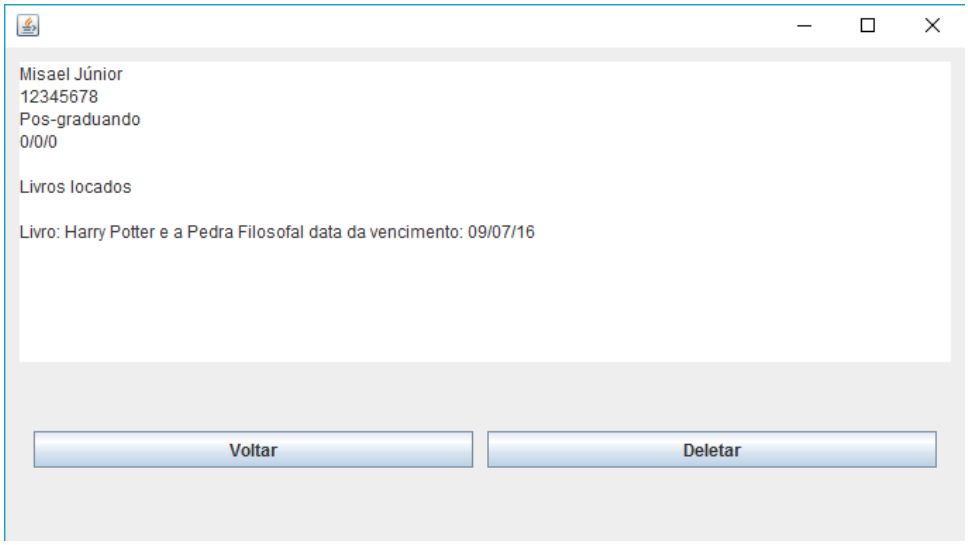
Gerou a linha Harry Potter e a Pedra Filosofal;12345;J K Rowling;5;* no arquivo Livros.txt.

Alugar um livro.

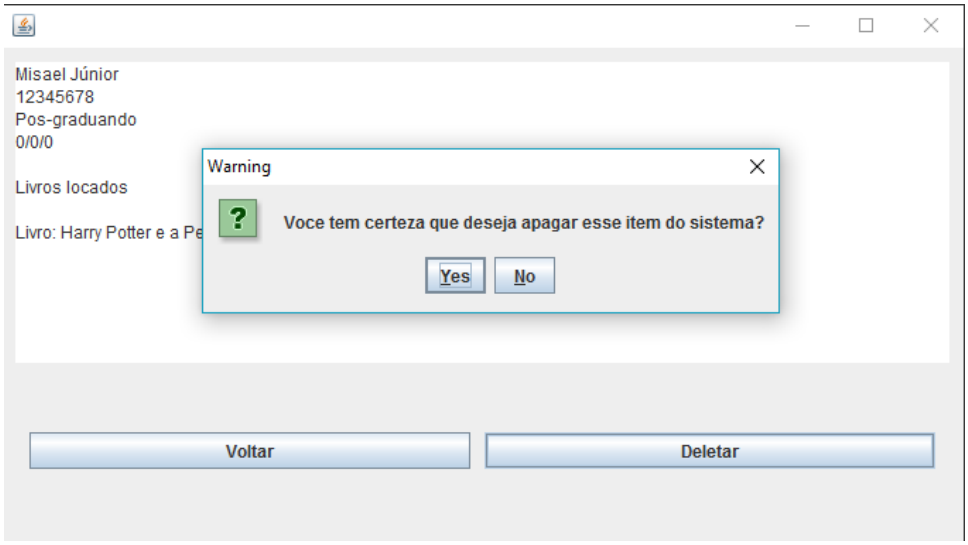


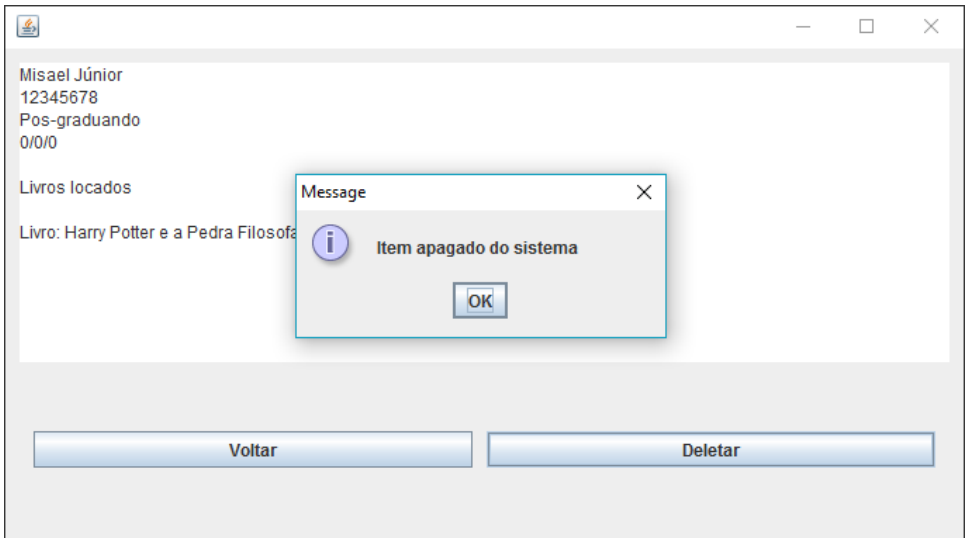
Mudou as linhas anteriores para "Misael Júnior;12345678;Pos-graduando;0/0/0;*;Harry Potter e a Pedra Filosofal;09/07/16" e "Harry Potter e a Pedra Filosofal;12345;J K Rowling;4;*;Misael Júnior;09/07/16" respectivamente em seus arquivos de origem.

Pesquisar um usuário.



Apagar usuário do sistema





Bibliotecas de destaque

Além das bibliotecas comuns utilizados na maioria das aplicações em Java, foram necessárias e utilizadas as seguintes bibliotecas de destaque:

`javax.swing` para a criação da interface gráfica.

`Java.util.Calendar` e `java.util.Date` para fazer as operações com receber a data atual do sistema data como calcular o período para a devolução de um livro, por exemplo.

`java.io` para operações de manipulação de arquivos (abrir e fechar, por exemplo).

`java.util.Scanner` para escrever e ler dos arquivos.

`Java.text` Para auxiliar na conversão entre os tipos *String* e *Date*

Restrições

A restrição mais notável é a dependência das operações por exatidão no input. Para alugar um livro, por exemplo, deve-se escrever o nome do livro e do usuário exatamente como estes estão salvos no sistema, sem margem de erro. Para ajudar nesse problema foi implementada a função de identificar usuários e livros por um código. Assim, é mais fácil identifica-los sem cometer erros de escrita.

Também se mostrou inoportuna a incapacidade de editar um usuário. Para editar as informações é necessário apagar o usuário anterior e criar um novo. Essa funcionalidade poderia ser implementada utilizando quase exclusivamente código já escrito porém não houve tempo devido às atividades de final de semestre.

Além disso, quando o problema apresenta problemas inesperados, os dados são corrompidos e cabe ao usuário desfazer através das funções do sistemas quaisquer erros nas informações causadas por eventualidades.

Problemas

Devido à vasta quantidade de afazeres, não foi possível corrigir os problemas e implementar as funções seguintes:

- Ao apagar um usuário não remove os livros alugados desse usuário, ou seja, eles são contabilizados como se ainda estivessem locados.
- Apesar dos métodos de comparação e escrita de datas estarem funcionais e ter sido implementada a função de checar se o usuário tem pendências, não foi implementada a função de calcular as penalidades para atrasos e as registrar no momento de devolução de livros atrasados.
- Bugs e problemas de caráter não constante nas funcionalidades surgem esporadicamente.

Requisito

Dentro das restrições acima, foram implementas as funcionalidades propostas. São elas:

Criar usuário (dos tipos professor, pós-graduando, funcionário e aluno)

- Deletar usuário
- Adicionar livro
- Remover livro
- Emprestar livro (se possível)
- Devolver livro
- Procurar usuário (e checar informações de empréstimos)
- Procurar livro (e checar disponibilidade)

Porém, é necessário apontar que para um sistema viável do ponto de vista comercial, essas restrições e problemas não poderiam ser toleradas. O sistema, como foi apontado, abre margens para inconveniências que, a priori, não deveriam ocorrer. Além disso, esses problemas não são reparáveis por usuários sem conhecimento de programação e do funcionamento interno do sistema.

Apesar dos métodos estarem desenvolvidos de maneira confiável, a junção deles no sistema completa mostrou problemas de execução que, até a data de entrega do projeto, o tornam comercialmente inviável.

Crescimento acadêmico

Para chegar ao resultado atual foram feitos muitos erros. Esse erros, porém, não foram vistos negativamente e tratados como acontecimentos a serem esquecidos. Eles ajudam a nos apontar para novas direções. Durante o desenvolvimento das primeiras versões do sistema, classes ambíguas foram criadas, métodos não suficientemente testados foram dados como prontos, lógicas confusas foram adotadas, pedaços de códigos prontos os quais eu não tinha domínio suficiente para operar foram adotados, dentre outros erros. Aprendendo sempre com cada nova versão, cada reiteração, cada reescrita de um pedaço de código, o projeto tomou e forma e, mais importante, eu, cada vez mais e sempre, me moldo como programador. Infelizmente não foi possível continuar lapidando, melhorando e implementando funcionalidades do projeto até o ponto que eu me sentisse confortável com o código e ele me representa como programador. Porém, todas as dificuldades que foram superadas me levaram a mais perto desse objetivo. A felicidade envolvida em consertar um *bug* depois de tanto tempo e tantos *System.ou.println();* e uma reservada para nós programadores. Por esses momentos de triunfo e motivação há engrandecimento acadêmico e pessoal na disciplina de ser programador mesmo em um projeto simples de faculdade e em meio a tantas outras atividades de fim de semestre.